Sentence Realization from Bag of Words

(Project 10)

Project Final Report

CSE 549 – Computational Biology

Anchal Agarwal – 108997912

Rishi Josan – 108996773

Aakrati Talati – 108996980

Approach 1: Greedy approach using Bigram-Model

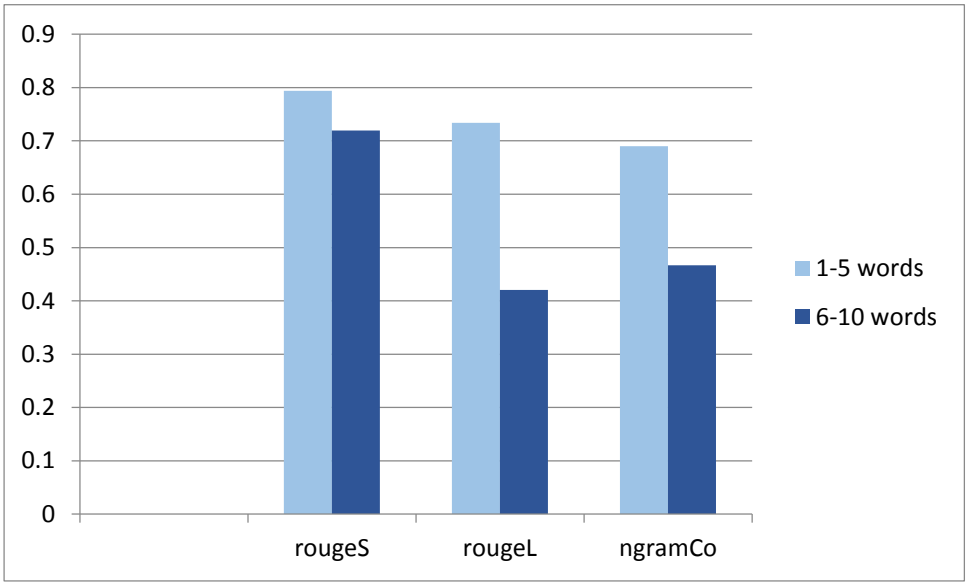
***Input**:* A bag of words.

***Algorithm**:* We used 2-gram model to generate a set of sentences from a given bag of words. We create an intermediate mXm matrix to solve this. The key idea is that this algorithm follows a greedy approach where the bigram with the highest probability is taken as the next word in the sentence.

Produce 'm' sentences considering each word as the first word and the sentence formed from it. The next word in the sentence is chosen by successively choosing the bigram with the highest probability. The sentence most likely to be correct will be the one with the overall highest probability.

Output

Output for different scoring methods for reuters corpus:



Why Bad?

The sentence structure is developed in a sequential manner considering the highest bigram probability of the occurrence of word1 after word2 and so on. As with all other greedy approaches, this solution fails to give the optimal solution as it just considers the local maximum for a bigram phrase rather than taking into account the global optimal structure.

Comparing the greedy approach with the Bigram MST approach:

Correct sentence: *Prior year earnings restated to reflect recapitalization plan*

Bag of Words: *['year', 'plan', 'restated', 'recapitalization', 'reflect', 'to', 'earnings', 'Prior']*

(Randomized input words)

Output Greedy approach: *Prior to reflect year plan earnings restated recapitalization:* score- *0.67*

Output Bigram approach: *Prior earnings restated year to reflect recapitalization plan :* score-*0.93*

The comparison above shows that the greedy approach is not efficient enough to give correct results in every case. For shorter sentences such as "*I want to eat food"* it gives perfect results.

Approach 2: Sub tree Type Language Model using Bigram

Motivation:

The greedy approach fails to consider a global structure of the sentence, Maximum Spanning tree on other hand develops a structure as a whole and does not construct the sentence sequentially. It considers the maximum weight edges between nodes (words) based on bigram probabilities.

Moreover, syntactic information was available using POS tagging, which incorporated some grammatical information about the structure of the sentence. It also improved the scores by nearly 7-10%. It was easier to represent it as a directed dependency tree.

The following parent-child relationship can be seen:

One-to-One: The probability of occurrence of a word after another word is high.
One-to-Many: The probability of occurrence of multiple words after a single word is equally likely.
Many-to-One: The probability of occurrence of a word after multiple words is equally likely.

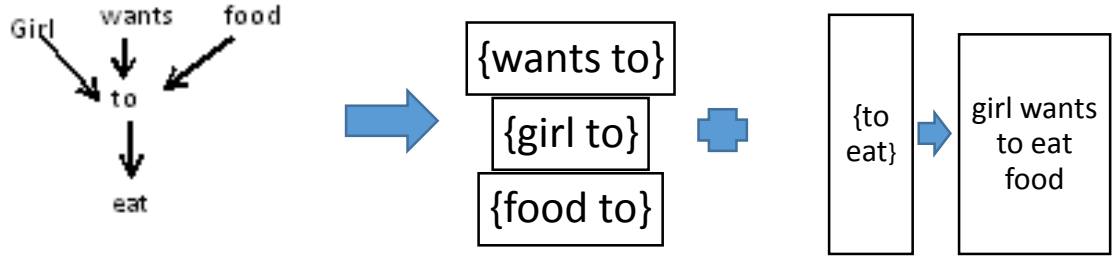
Key Idea:

The key idea is to **combine greedy approach and brute force** approach of generating n! permutations. However, the number of permutations sent to the scorer in this approach are **reduced to n^2**. We came across sentences that formed the above relationships and could not be resolved sequentially using greedy approach.

For example:

Correct Sentence: *"Girl wants to eat food"*

MST generated:



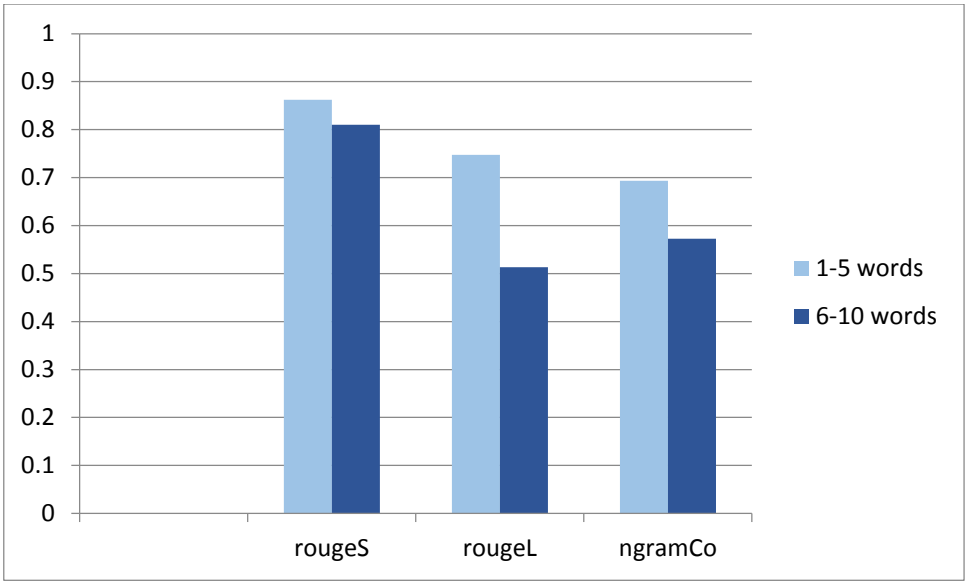
In the above tree, *'to'* has many-to-one relationship. First, the algorithm will follow a greedy approach to combine a phrase with highest probability, here it will be *'wants to eat'* and then it will create permutation of sentences with remaining phrases, i.e. *'girl'* , *'food' and 'wants to eat'*. In a nutshell, the approach starts with the greedy approach and when it fails to combine any more phrases it moves to permuting the remaining words/phrases.

Moreover, if followed only greedy approach, then it would have sequentially combined the words based on bigram probabilities without considering the global possibilities.

Output for greedy: *"food wants to eat Girl"* :)

Output

Output for different scoring methods for Reuters corpus:



Why Bad?

Although, we have the global optimal structure still we think we are not able to fully exploit the information available. The quality of sentence increases as we move to higher order n-gram models. Hence, we consider trigrams for better quality of sentences as compared to those obtained using bigrams.

Structure followed for the resolution of highest probability occurring phrases is:

Trigrams probability calculated as: *[{Parent}, {Node}, {child}]*

Bigrams probability calculated as: *[{Parent}, {Node}]: [{Node}, {Child}]*

Comparing the Trigram approach with the Bigram MST approach:

Correct sentence: *Prior year earnings restated to reflect recapitalization plan*

Bag of Words: *['year', 'plan', 'restated', 'recapitalization', 'reflect', 'to', 'earnings', 'Prior']*

(Randomized input words)

Output Trigram approach: *Prior year earnings restated to reflect recapitalization plan:* score-*1.0*

Output Bigram approach: *Prior earnings restated year to reflect recapitalization plan:* score-*0.93*

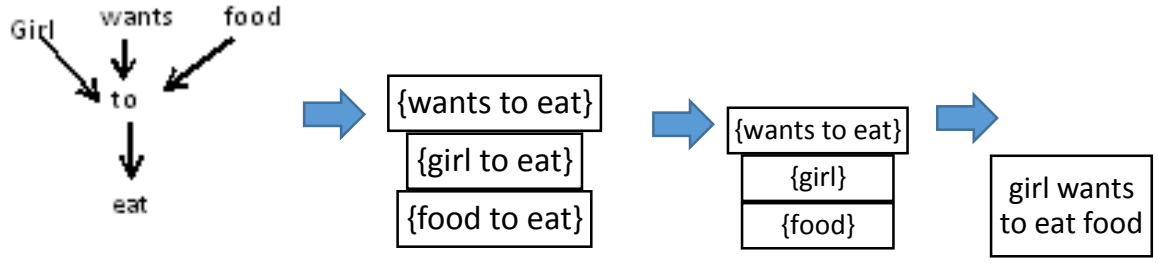
Improvement over Approach 2: Sub tree Type Language Model using Trigram

Considering trigram probability gave more accurate and quality sentences.

For example:

Correct Sentence: *"Girl wants to eat food"*

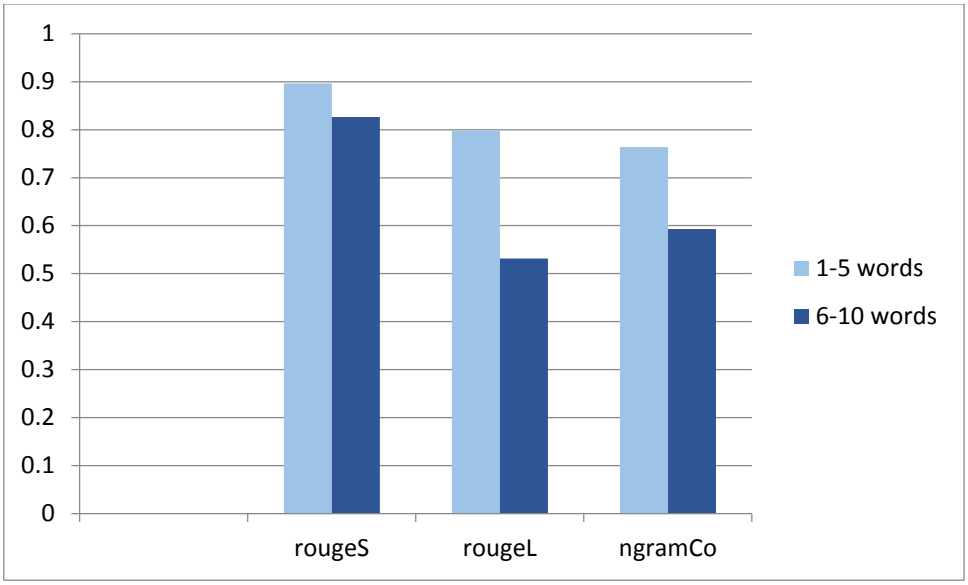
MST generated:



This approach builds on the previous bigram approach. In the example above, *'to'* has many-to-one relationship. Here, it will consider the phrase which has maximum probability for *'wants to eat', 'girl to eat' and 'food to eat'. And* then it will create permutation of sentences with remaining words, here *'girl'* and *'food'*.

Output

Output for different scoring methods for Reuters corpus:



Noun Phrase Chunking

We implemented Noun Phrase Chunking as a means to reduce the number of permutations by exploiting grammatical features of the English Language. We chose the noun phrase as it is not just part of both the subject and object in a sentence, but is also a component of the verb phrase. As such, we envisioned that finding the top three noun phrases in a sentence would help us greatly reduce the permutations.

We were successful in generating Noun Phrases from rules that were discussed in the progress report. But when we integrated NP chunking with our core algorithm, the results weren't very good.

Sentences usually have a couple of nouns and determinants. Even though we were successful in generating all possible Noun Phrases, we were not successful in choosing the right one to proceed with.

For example:

For the sentence *"the little yellow dog barked at the blue greedy cat"*.

The top noun phrases we get are *"the blue little yellow cat"* and *"the greedy dog"*. Although these noun phrases are correct, these are not what the original sentence comprised of.

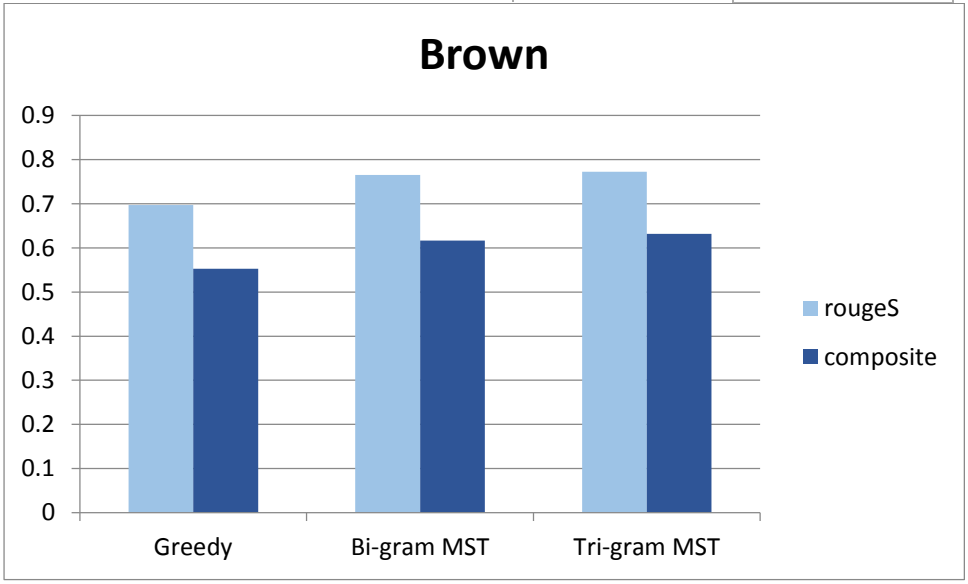
Noun phrase chunking works well for short sentences (5-6 words) which have only a single noun. Therefore we did not integrate Noun Phrase chunking in our final model.

Comparison of Mean Scores for different approaches

Looking at the rougeS and composite (average of rougeL, rougeS and ngram co-occurrence) scores, we see that our final approach is in fact the best. The following statistics chart these scores for all three test sets independently.

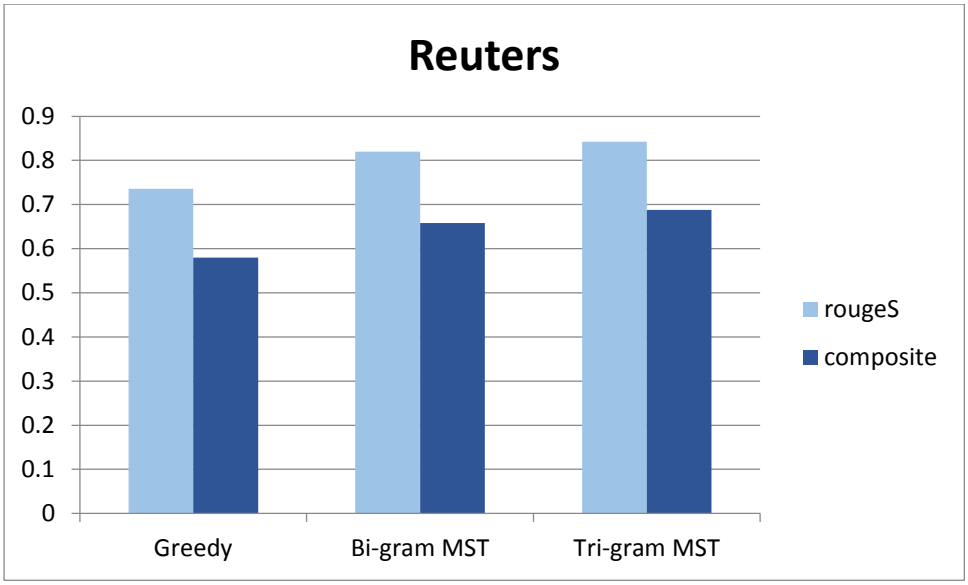
Brown

	rougeS	composite
Approach 1 : Greedy	0.697438893	0.553067712
Approach 2 : Bigram MST	0.76529535	0.616499698
Approach 3 : Trigram MST	0.772442778	0.632211671



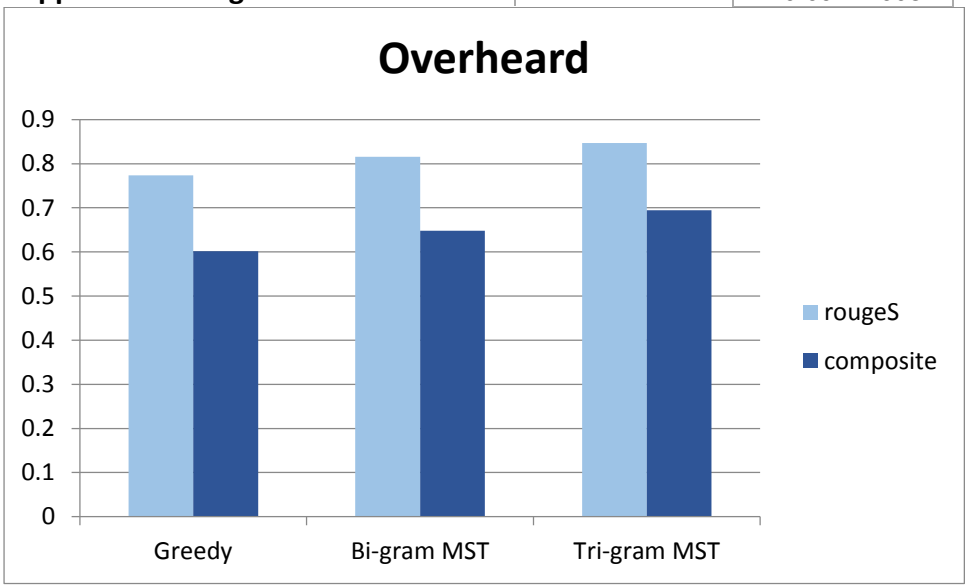
Reuters

	rougeS	composite
Approach 1 : Greedy	0.735667845	0.579696634
Approach 2 : Bigram MST	0.819875473	0.657992986
Approach 3 : Trigram MST	0.84212214	0.688195523



Overheard

	rougeS	composite
Approach 1 : Greedy	0.773584465	0.60126428
Approach 2 : Bigram MST	0.815728111	0.648130921
Approach 3 : Trigram MST	0.84683102	0.694176051



Limitations

We have not handled punctuation in its entirety for our approach. The test set has been constructed keeping this in mind

Appendix: Test Results

Please Flip Over